

# A diffuse domotic control powered by a networked intelligence

Luca Ferrari\* and Gian Luca Galliani\*

\*Department of Computer Science

University of Milano, Via Comelico 39, 20135- Milan, Italy

Email: {luca.ferrari6,gianluca.galliani}@studenti.unimi.it

## Abstract

We describe a diffuse control system for household appliances rooted in an Internet of Thing network empowered by a cognitive system. The key idea is that these appliances constitute an ecosystem populated by a plenty of devices with common features, yet called to satisfy in an almost repetitive way needs that may be very diversified, depending on the user preferences. This calls for a network putting them in connection and a cognitive system that is capable to interpret the user requests and translate them into instructions to be transmitted to the appliances. This in turn requires a proper architecture and efficient protocols for connecting the appliances to the network, as well as robust algorithms that concretely challenge cognitive and connectionist theories to produce the instructions ruling the appliances. We discuss both aspects from a design perspective and exhibit a mockup where connections and algorithms are implemented.

## I. INTRODUCTION

Control theory, as an offspring of cybernetics, is deeply rooted in the concept of feedback [1] since the forties of the previous century. In the same period of time a different notion of control in terms of self adapting systems emerged with the first modern hypotheses of the brain computing facilities within the connectionist framework [2]. These two ways of controlling a dynamic system ran in parallel up until the nineties, when hybrid control systems began exploiting synergies by conventional controls and by either recurrent neural networks [3] or reinforcement learning algorithms [4]. The Internet of Things paradigm suggests a renewed synergy between the two approaches. Namely, the network connecting things on the Web enjoys many properties of a neural network in terms of both connectivity and elementariness of the messages normally exchanged between the devices and their huge number. However, rather than distributed as in the connectionist paradigm, computations are expected to be more efficient if they are performed in a centralized way, yet exploiting the capillarity of the information the network may bring them – hence we call it *networked intelligence*. *Per se*, the machine where computations are done is immaterial, so that we may assume them to be carried out (and possibly distributed too) everywhere in the cloud. However, as with the multilayer perceptron [5], the information management is dealt with better through a hierarchical architecture than through a distributed one.

This is the idea we pursue in our ecosystem. Namely, we are devising a system constituted by an ensemble of household appliances ruled by a social network which is from time to time committed by a user to operate them optimally with respect to a given task. For instance, the user asks the network to have trousers perfectly washed by his washing machine. In reply, the network sends directly to the machine a sequence of instructions, call it *recipe*, such as “charge water, heat water to 35 degrees”, etc., which drive the machine to carry out a perfect washing. On the one hand this a typical procedure we are used to expect from our personal devices. On the other one, the way of implementing the procedure is rather hard. In a extreme synthesis we need:

- 1) electronics allowing the network to communicate with the machine, possibly overriding its microcontroller logic;
- 2) a logic able to produce recipes that are optimal in respect to many criteria, from user preferences all the way to ecological goals such as water or electricity saving. The logic must learn how to reach these objectives from the feedback coming both from devices and from users. Hence it is a cognitive system.
- 3) an architecture and protocols vehiculating signals between devices and the networked intelligence in a safe and efficient way.

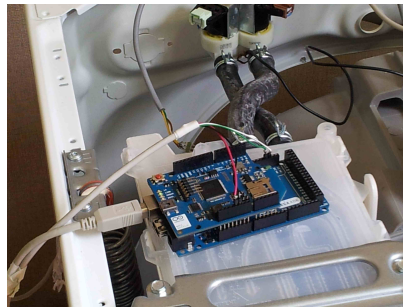
The good news is that the proposed ecosystem has some appeal, so that it got funded by the European Community with a horizon of 30 months (project Social&Smart (*SandS*)-<http://www.sands-project.eu/>). In this paper I describe an early mockup where instruction and signal dispatchings are enabled by proper circuits.

## II. AN ENABLING MOCKUP

We have set up a first mockup in the Computer Science Department labs of University of Milano. From a purely operational perspective, it consists of two - three white goods wifi connected to Internet, each with



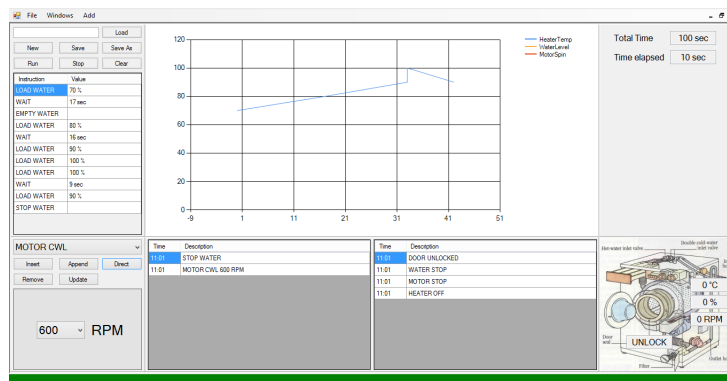
**Fig. 1:** A early SandS mockup.



**Fig. 2:** The Arduino interface.

its own Arduino board, in order to execute recipes and send back status signals. For the moment only a washing machine (*wm*) is being tested (see Fig. 1). We use an Arduino MEGA ADK board, an Arduino WIFI SHIELD (see Fig. 2) and a Sitecom wifi router connected to the university network.

On this hardware we implement a communication protocol solving, albeit in a preliminary way, problems both of security and of exact message addressing from middleware to appliance and back, within a MQTT-like service [6] that is sketched in a next section. The protocol is event driven, so messages are sent only if state variables change value. The appliance detection is realized in plug&play mode. The management of the appliance, in terms of specification and recipe requests, is done via a browser on any mobile or fixed device. The appliance is located on the ground floor of the Computer Science Department, while the console is on the third floor of the same building. A distance of around 300 m is covered by ethernet wiring, while the last 10 m (including two concrete walls) are gapped via wifi connection. A sketch of the parameters managed by the browser is reported in fig. 3. In the following we propose early considerations on the various parts of the mockup.



**Fig. 3:** A screenshot of the washing machine monitor.

#### A. Computational requirements to the interfacing board

Quite simply, we ask to the Arduino board to act as a transducer: in input an instruction to the wm actuators, in output the corresponding signal to be transmitted to the wm microcontroller. From this perspective, the signals overwrite the native microcontroller firmware with a set of commands that change over time. Instructions are at moment generated by a remote computer in the role of Domestic Intelligence (DI), as a temporarily substitute of the networked intelligence. Thus the microcontroller receives one command at a time consisting of the actuator ID and the value of the parameter to be fixed, where the time is decided in principle by the DI. On the one hand this sort of unitary code (one parameter per actuator) requires common tricks to manage multiparameter actuators. On the other hand, we distinguish two time scales, i.e. a time granularity triggering the shift between scales. The general philosophy is: a sequence of instructions, each lasting more than the time granularity, is dispatched at the proper time by the DI. Vice-versa, sequences of instructions lasting less than the time granularity are encoded into a single instruction that is interpreted and sequenced directly by the Arduino board. This requires a finite automaton to reside in the board, both to carry out the above sequencing and to manage a series of security checks to avoid plant damages and logical inconsistencies. As a matter of fact, even one shot instructions such as “heat the water” requires a set of controls – e.g. the water level to prevent the system from burning out in case of an empty drum – which are managed locally by Arduino. However, as to granularity, this instruction falls in the first category. One example of the second category is the alternating running of the wm motor during the pre-wash phase.

This is the basic situation with regard to normal running. In addition, the board is called upon to supervise an anomalous running at two levels – warning and alarm – to which different standby or shutdown sequences may follow as a consequence of the recognized drawback.

Thus the interfacing board is required for computational power to store the instructions to be decoded locally and to manage local time during the implementation of these instructions. For these purposes, the Arduino capacities (RAM 256 KB, processor ATMEGA2570) prove to be sufficient.

#### B. Communication requirements to the interfacing board

The communication bandwidth necessary for the mockup is normally very low. It increases relatively when the appliance is woken-up by a recipe execution request. The communication is stroked by the *keep alive* message sent by Arduino to the DI. Thus, every 5 seconds, the board waits for instructions by the DI. In the current debugging phase the board sends the appliance status (water temperature and level, spin rpm) as a more informative payload. In a greater detail, the message from board to DI is structured as follows, as a further simplification of MQTT scheme:

---

```
[CMD,EVN,VAN,VALUE,CHK]
```

```
[ = start of message
```

```
CMD = command.
```

```
EVN = (progressive) Event number.
```

```
VAN = Var number (indexing the boling up variables).
```

```
VALUE = Var value (type:Byte, Int, Long o String).
```

```
CHK = Checksum
```

```
] = end of message
```

---

Conversely, the DI sends instructions to the board. We index each instruction with a sequential number within a recipe ID. This indexing may prove suitable for both debugging and appliance quality control purposes. The message format is analogous, with internal event number mating with the one of the sent event for a correct reckoning of the queues.

---

```
[CMD,EVN,VAN,VALUE,CHK]
```

```
[ = start of message
```

```
CMD = command.
```

```
EVN = Event number (the same of the message which is the answer to).
```

```
VAN = Var number.
```

```
VALUE = Var value.
```

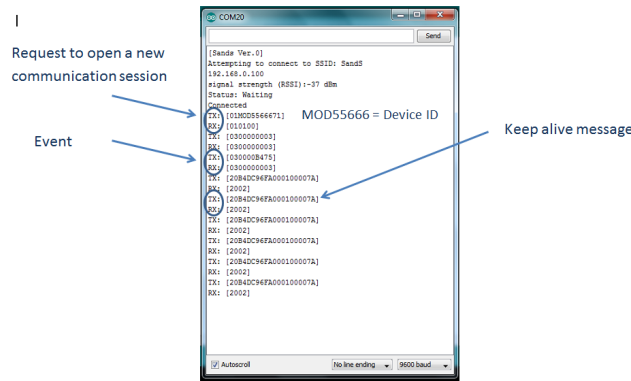
```
CHK = Checksum
```

```
] = end of message
```

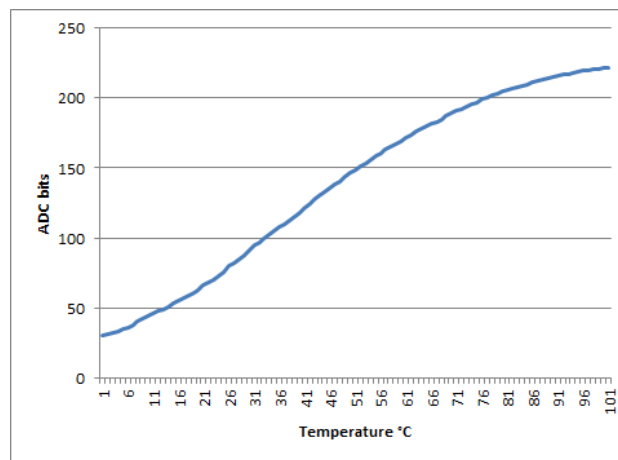
---

Once the connection is stated between DI and board, it proceeds in full duplex mode. The connection is open and maintained by the appliance through the *keep alive* messages. In Fig. 4 we can see a segment of the conversation between appliance and DI, as collected by the Arduino debugger.

The time granularity of the recipe transmission takes into account various idle times. Besides normal messages related to the recipe execution, both communication addressees may send overriding messages concerning alarm status and various kinds of shutdown/standby commands.



**Fig. 4:** A screenshot of the Arduino debugger.



**Fig. 5:** A calibration curve.

### *C. Degrees of freedom in overwriting the microcontroller software*

In principle the Arduino Interface is an open-hardware/open software device, so that any person may install it on his/her white good without requiring any license. However, its implementation requires a set of weld junctions and software for overwriting the appliance microcontroller that are specific to the appliance in hand. A further step in the mockup implementation will be to extend the software standardization to a maximum. But overwriting the microcontroller is not an easy task. Rather, we may expect the emergence of small business third parties – for instance, free lance professionals or white good stores with modern selling strategies – which are specialized in this task. For instance, Fig. 5 reports a typical calibration curve relating the water temperature level in the drum and the electrical signal transmitted to the board. Nevertheless, the obvious expectation is that appliances' manufacturers will pursue their own business interests and support this new paradigm of appliance usage by embedding their products with the necessary electronics.

In this early implementation, we played the role of networked intelligence substitute by featuring a few recipes by ourselves. This gave us the opportunity to appreciate the degrees of freedom of this operation and the optimality criteria we may pursue as a counterpart. Namely we jointly considered the following goals: 1) washing efficacy, 2) energy consumption, 3) water consumption and 4) environmental pollution.

## III. CONCLUSIONS

While the project is still at an initial stage, we have come to see the high value it offers in terms both of user convenience and of environmental advantage. For instance, questions re the benefit and ecological profitability of using a bio soap product or reducing the amount of wash water become theoretical opinions no longer. Though the concrete operations necessary for realizing the mockup delineate a scenario where each single user may implement her/his SandS terminal on home appliances with some technical help, the main way to implement our paradigm passes through a both moral and economical suasion to convince the manufacturers that social appliances are more efficient and rewarding. To achieve this goal SandS project

will realize the above discussed architecture in order to set up an initial social network of eahoukers where the benefits of the paradigm will be tossed in concrete by a thousand members. These benefits and the members enjoying them will be the authentic promoters of the SandS ecosystem, again all on the spirit of exploiting actual facts besides sentences.

#### REFERENCES

- [1] N. Wiener, *Cybernetics, Or Control and Communication in the Animal and the Machine*. New York: Wiley, 1948.
- [2] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.
- [3] C.-C. Ku and K. Y. Lee, "Diagonal recurrent neural networks for dynamic systems control," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 6, no. 1, pp. 144–156, 1995.
- [4] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.
- [5] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1994.
- [6] "Mqtt standard." [Online]. Available: <http://mqtt.org/>